Build Automation for Mobile

or

How to Deliver Quality Apps Continuously

Angelo Rüggeberg
„Publishing your App should not be painfull“
Angelo Rüggeberg

„Code Quality Matters“
Angelo Rüggeberg
Who am I?

- Mobile Development
- Android
- iOS
- Backend
- Cloud Infrastructure

http://plus.google.com/+AngeloRüggeberg

http://s3xy4ngyc.github.io/  s3xy4ngyc
Release Process
Current Situation
Release Process
Current Situation

WHAT IF I TOLD YOU
MAKING A RELEASE DOES NOT HAVE TO BE PAINFULL
Everything you have to do afterwards is

git push
Release Process
the future
Build Automation
What will we be using

jenkins
Test Driven Development
Git
Beta Distribution
Gradle
Nightly Versions
Build Automation

Benefits

• New Features are Live ASAP
  • As Soon As Pushed
• Only Stable and Tested Builds can be Pushed
  • No more unstable Releases
  • Less Human Testing
• No more Sharing of Sensitive Data
  • Keystores
  • Logins
Jenkins is an award-winning application that monitors executions of repeated jobs, such as:

- Building/testing software projects continuously
- Monitoring executions of externally-run jobs

Sample Build Pipeline:

- compile
- lint
- Tests
- sonar
- monkey
- runner
- distribute
Build Automation

Build Pipeline

- Distribution will only happen if all previous steps succeed
- Tests, code analysis, etc. can run parallel
- Step by step enables failing fast
  - Resource hungry jobs like Monkey Runner will be triggered only if needed
Clone Workspace SCM Plugin
  › Shared Workspaces may have locks
    › https://wiki.jenkins-ci.org/display/JENKINS/Clone+Workspace+SCM+Plugin

Polling Sucks
  › use Git Hooks!

Do not run on a Small Virtual Machine
  › Android Emulator needs some Hardware
    › Intel HAX strongly Recommended!
Jenkins
Build Monitor Plugin
Jenkins
Monitoring Plugins

my team  [build # 1542]  zynxhealth

**build**
- 6 mins
- Last ran 2 mins ago
- Stable

**deploy**
- 2 mins
- Last ran 21 mins ago
- Pass

**tests**
- 46 mins
- Last ran 3.12 hours ago
- 6% failed

---

**mature**
- 1.3 hours
- Last ran 2.4 hours ago
- Fail

**regression**
- 57 mins
- Last ran 4.5 hours ago
- Pass
Build Automation

gradle

- Best of ANT / Maven / GANT / Ivy
- Based on Groovy Scripts
- Expressive DSL
- (Flexible) Convention over Configuration
- Easy to extend

„Make the impossible possible, make the possible easy and make the easy elegant.”
- Staging, Dev and Live Packages
  - On Your Test Device no Constant Uninstalling
  - Adjustable Configurations without Code Changes

- Why?

- Reproducible builds, easy setup
productFlavors {
  live {
    android.defaultConfig.applicationId
    versionName = android.defaultConfig.versionName
  }
  stage {
    android.defaultConfig.applicationId + ".stage"
    versionName = android.defaultConfig.versionName + "-STAGE"
  }
  dev {
    android.defaultConfig.applicationId + ".dev"
    versionName = android.defaultConfig.versionName + "-DEV#$\{gitSha()\}"
  }
}

sourceSets {
  live {
    java.srcDirs = ['src/live/java', 'src/main/java']
  }
  dev {
    java.srcDirs = ['src/dev/java', 'src/main/java']
  }
  stage {
    java.srcDirs = ['src(stage/java', 'src/main/java']
  }
}
Project Setup Flavoring

- Same Folder Structure for every Flavor
- Flavor Specific files Overwrite Files from Main
- Only Overwrite What your Flavor Needs
- Flavor Only Files Possible
Example Flavor Specific Manifest

• Only Provider defined

• Everything else comes from main/AndroidManifest.xml
Build Automation

- Branching
- Opensource
- Integrated and Extended Implementations
Git Flow Branching Model

source: [www.atlassian.com](http://www.atlassian.com)
Jenkins

Git Flow and Jenkins Examples

- Staging
  - Hook on develop Branch
- Nightly
  - Time Based Build (eg once a day at 0 AM)
  - builds newest feature Branch
- Live
  - Only Builds release Tags
  - Release Notes are Tag Message
Beta Test Distribution
Examples

- Google Play
  - Alpha / Channel
  - Works on every Device that has The Playstore
- Crashlytics
  - Distribution Groups & User Level
  - gradle Integration
  - free
- HokeyApp
  - Comercial
Unit Testing
WHY?

Testing - Do it
Unit Testing
WHY?

WORKED FINE IN DEV

OPS PROBLEM NOW
Unit Testing
WHY?

- Code Quality Matters!
- Insurance everything still works
  - As it should be working
- Refactoring made easy
- Unittests are somehow a Functional Documentation
Unit Testing
Some Tipps:

- Tests first!
- Simplest and dumbest way to implement
- Test Orientation Changes
- Test State Changes
  - Lock Screen
  - Incoming calls
  - etc.
- Check Code Coverage
Test Frameworks
Unit Tests

- Roboelectric
  - No Emulator Needed
  - No Mocking Frameworks Needed
  - API Level Emulation only up to 18 (currently)

- Robotium
  - Silenium for Android
  - User Scenarios

- Default Android Testing Framework
Test Frameworks

UI Testing

- Espresso
  - Ui Testing
- Double Espresso
  - Espresso Gradle Port
- Ui Automator
  - Android Default UI Testing Framework
robocode {
    include '**/Test.class'
    exclude '**/espresso/**/*.class'
}

dependencies {
    repositories {
        jcenter()
        mavenCentral()
    }
    // Espresso
    androidTestCompile files('lib/espresso-1.1.jar')
    androidTestCompile files('lib/testrunner-1.1.jar')
    androidTestCompile files('lib/testrunner-runtime-1.1.jar')
    androidTestCompile 'com.google.guava:guava:17.0'
    androidTestCompile 'com.squareup.dagger:dagger:1.1.0'
    androidTestCompile 'org.hamcrest:hamcrest-integration:1.1'
    androidTestCompile 'org.hamcrest:hamcrest-core:1.1'
    androidTestCompile 'org.hamcrest:hamcrest-library:1.1'
    androidTestCompile('junit:junit:4.11') {
        exclude module: 'hamcrest-core'
    }
    androidTestCompile('org.robolectric:robolectric:2.3') {
        exclude module: 'classworlds'
        exclude module: 'commons-logging'
        exclude module: 'httpclient'
        exclude module: 'maven-artifact'
        exclude module: 'maven-artifact-manager'
        exclude module: 'maven-error-diagnostics'
        exclude module: 'maven-model'
        exclude module: 'maven-project'
        exclude module: 'maven-settings'
        exclude module: 'plexus-container-default'
        exclude module: 'plexus-interpolation'
        exclude module: 'plexus-utils'
        exclude module: 'wagon-file'
        exclude module: 'wagon-http-lightweight'
        exclude module: 'wagon-provider-api'
    }
    compile 'com.squareup.fest:fest-android:1.0.8'
    compile 'com.android.support:support-v13:20.0.0'
    compile 'com.jakewharton:butterknife:5.1.2'
Test Frameworks
Device Testing

Distributing instrumentation tests

http://square.github.io/spoon/
Test Frameworks
Device Testing

Spoon Sample App
45 tests run across 5 devices with 30 passing and 15 failing in 2 minutes, 33 seconds at 2013-01-17 06:04 PM

DROID BIONIC
HTC One V
Nexus 4
Nexus 7
Nexus S
Test Frameworks
Device Testing

Make Some Salad, It Is Healthy
Ran on 5 devices with 4 passing and 1 failing in an average of 13 seconds

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTC One V</td>
<td></td>
</tr>
<tr>
<td>Nexus 7</td>
<td></td>
</tr>
<tr>
<td>Nexus 4</td>
<td></td>
</tr>
<tr>
<td>DROID BIONIC</td>
<td></td>
</tr>
<tr>
<td>Nexus S</td>
<td>Failed</td>
</tr>
</tbody>
</table>

junit.framework.AssertionFailedError: No TextView with text Next is found!
Test Frameworks

Device Testing
Build Automation
Measure Code Quality

- Measure Code Quality over Time
  - Duplicated code
- Coding standards
- Unit tests
- Complex code
- Potential bugs
- Comments
- Design and architecture
Test Frameworks

Code Coverage

- Cobertura
- EMMA
- Clover
- Jacoco

Offline bytecode instrumentation
Offline bytecode instrumentation
Source code instrumentation
On-the-fly bytecode instrumentation
Test Frameworks

Code Coverage

Unit Tests Coverage
- 53.4% coverage
- 81.8% line coverage
- 29.3% branch coverage

Unit test success
- 100.0%
- 0 failures
- 0 errors
- 1,588 tests
- 14.2 sec

Integration Tests Coverage
- 42.3% coverage
- 70.7% line coverage
- 18.1% branch coverage

Overall Coverage
- 61.7%
- 91.3% line coverage
- 34.6% branch coverage
Build Automation
Static Code Analysis

PMD
DON'T SHOOT THE MESSENGER

Find Bugs

CODENARC
Less Bugs Better Code
Build Automation
Static Code Analysis

```groovy
apply plugin: 'findbugs'
apply plugin: 'pmd'

task findbugs (type: FindBugs, dependsOn: assembleDebug) {
    description 'Run findbugs'
    group 'verification'
    classes = fileTree('build/classes/debug/')
    source = fileTree('src/main/java')
    classpath = files(project.configurations.compile.asPath)
    effort = 'max'
    excludeFilter = file('.config/findbugs/exclude.xml')

    reports {
        xml.enabled = false
        html.enabled = true
    }
}

task pmd (type: Pmd, dependsOn: assembleDebug) {
    description 'Run pmd'
    group 'verification'
    ruleSets = ["basic", "braces", "strings", "design", "unusedcode"]
    source = fileTree('src/main/java')

    reports {
        xml.enabled = false
        html.enabled = true
    }
}

check.doLast {
    project.tasks.getByName("findbugs").execute()
    project.tasks.getByName("pmd").execute()
}
```
Final Thoughts

- Testing your Application makes your life more Stress free
  - Code Quality
  - Less Production Crashes
- Don’t develop Stuff that should not go live
  - DONE = RELEASED
- Use Gradle Flavors!
Release Process
the future
>> /etc/